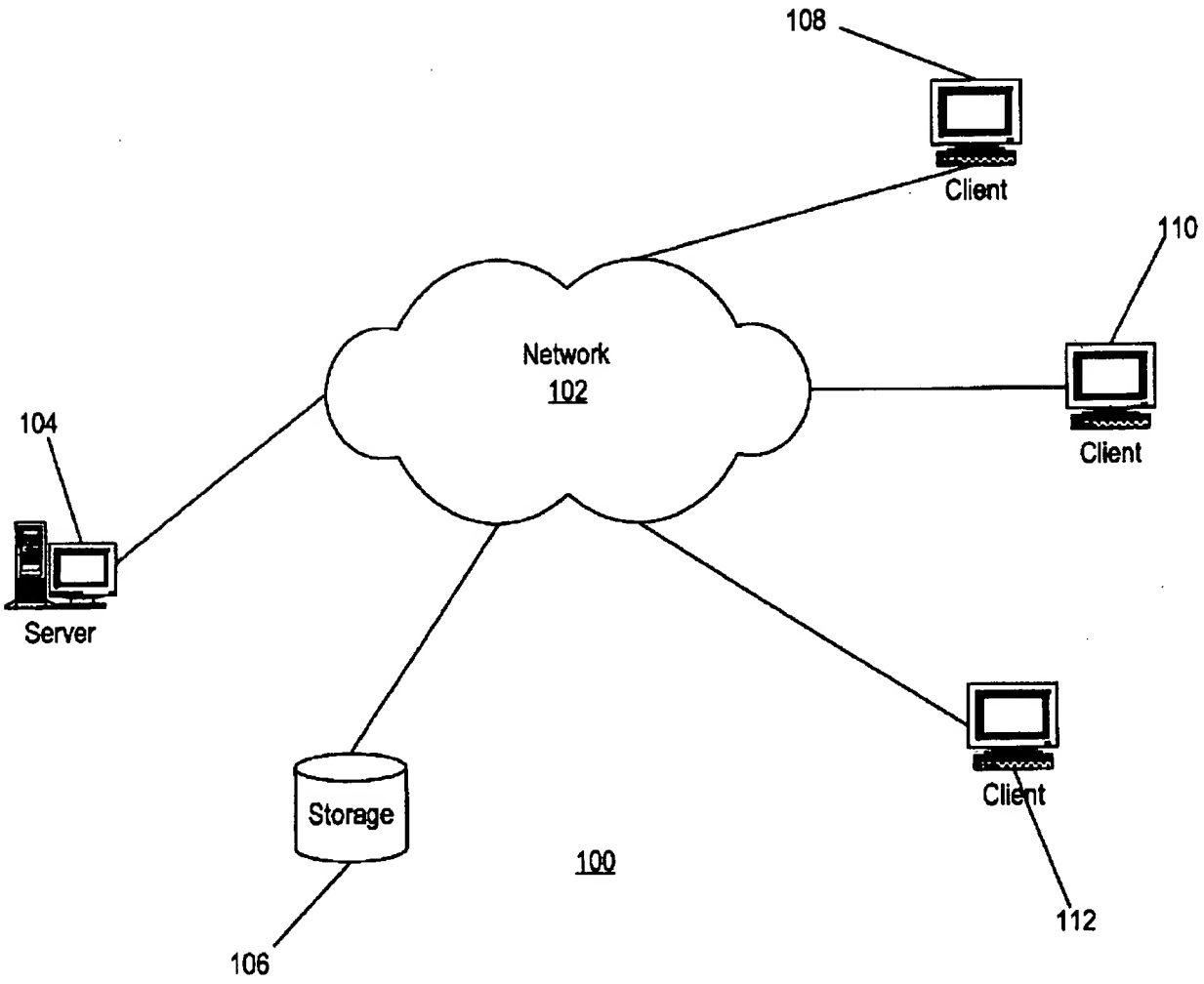
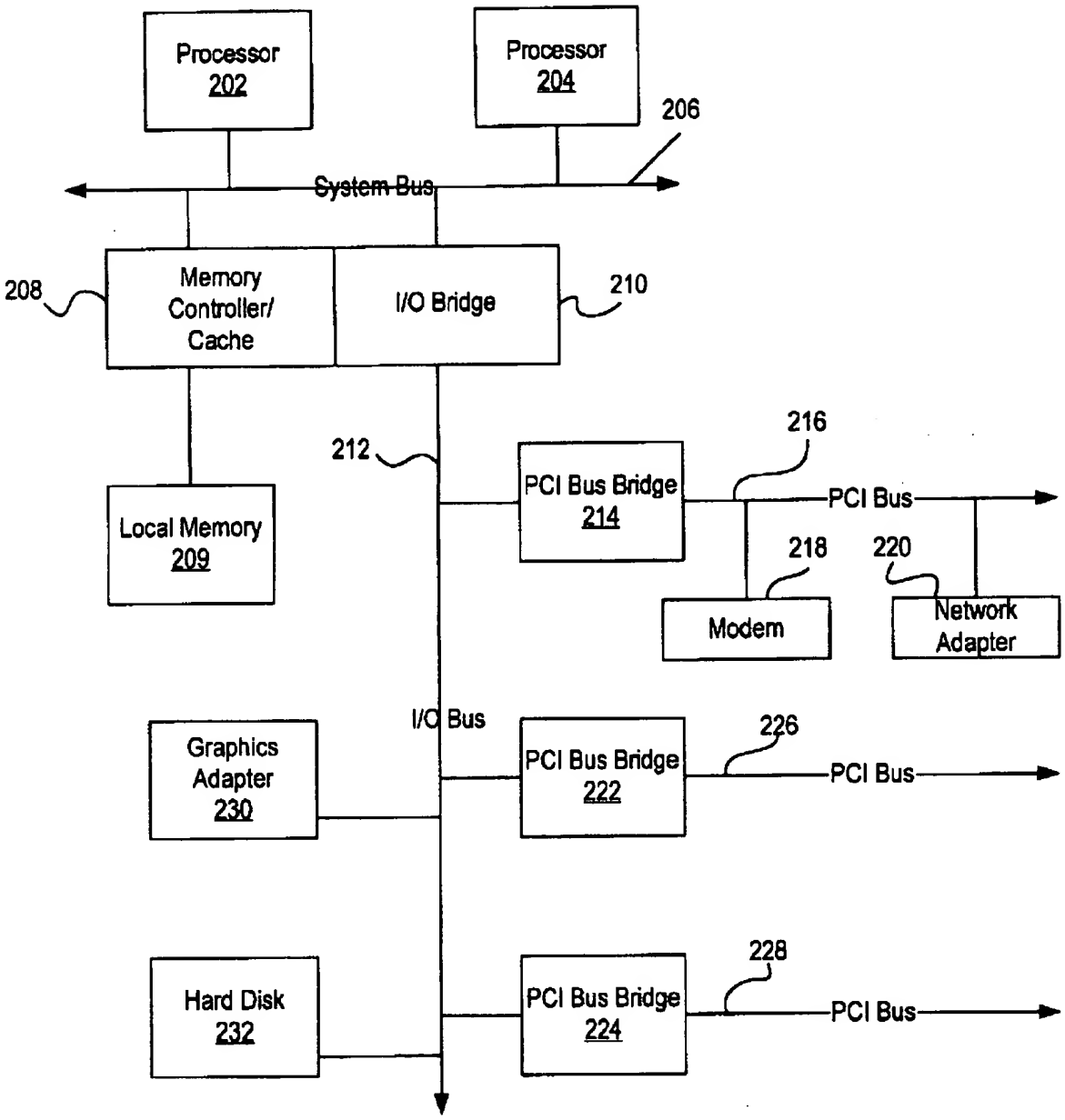
EK190671357US

Figure 1

AUS990853

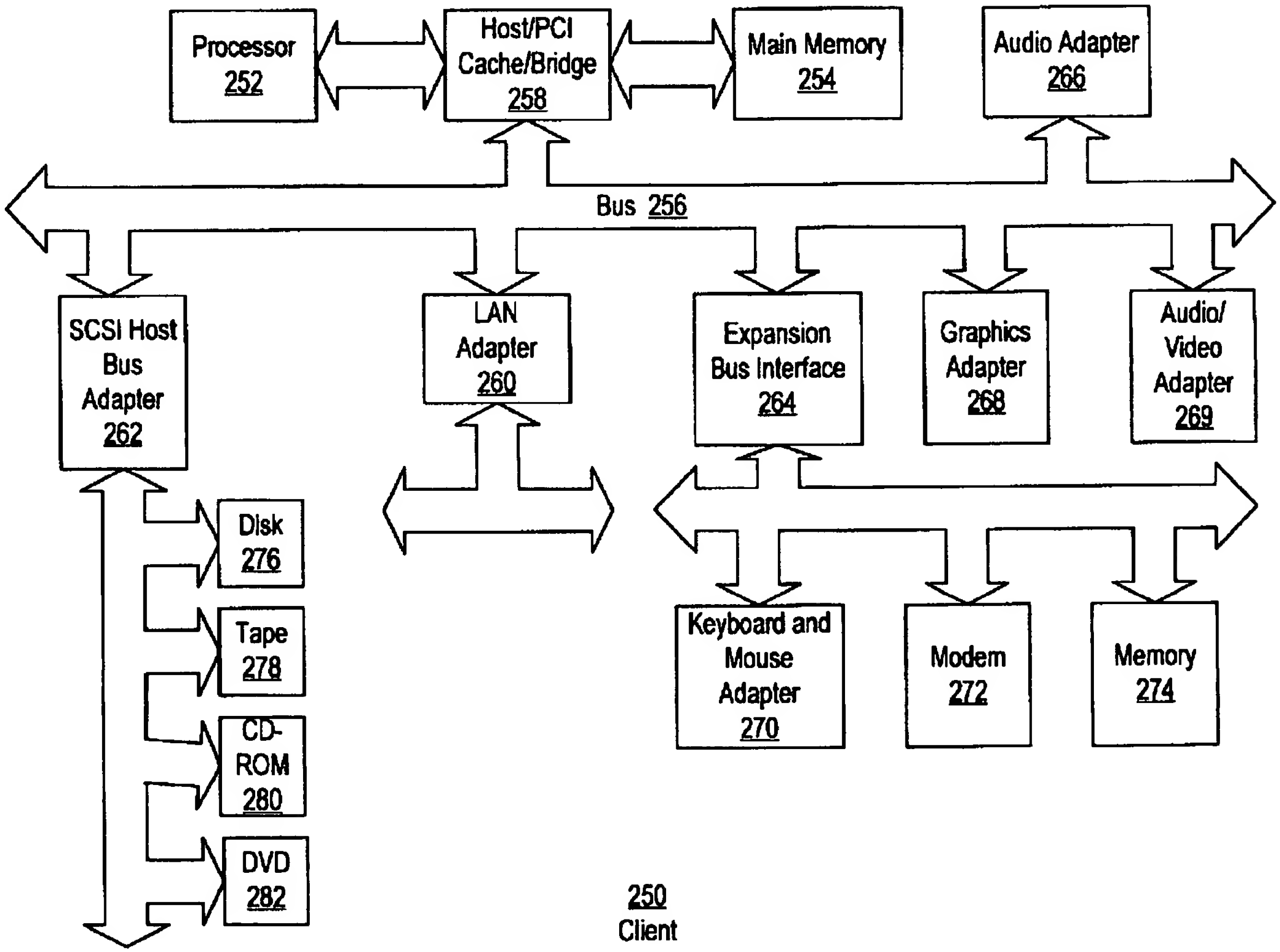




200

Figure 2A

AUS990853



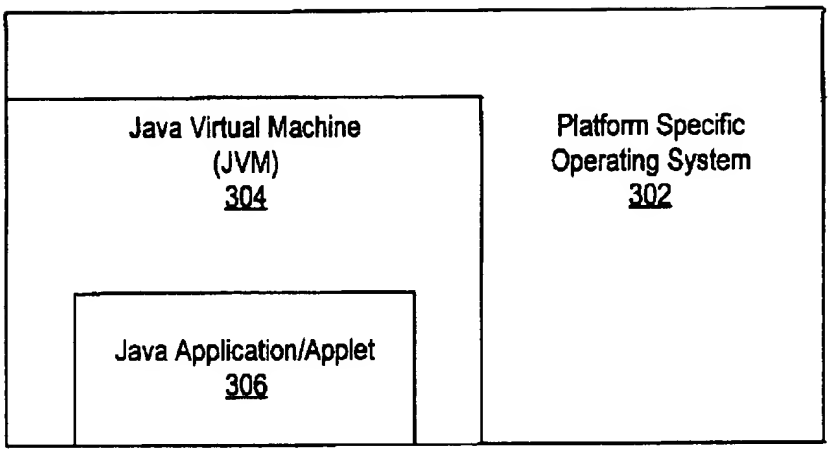
250
Client

Figure 2B

AUS990853

Figure 3A

AUS990853



300

Figure 3B

AUS990853US1

350

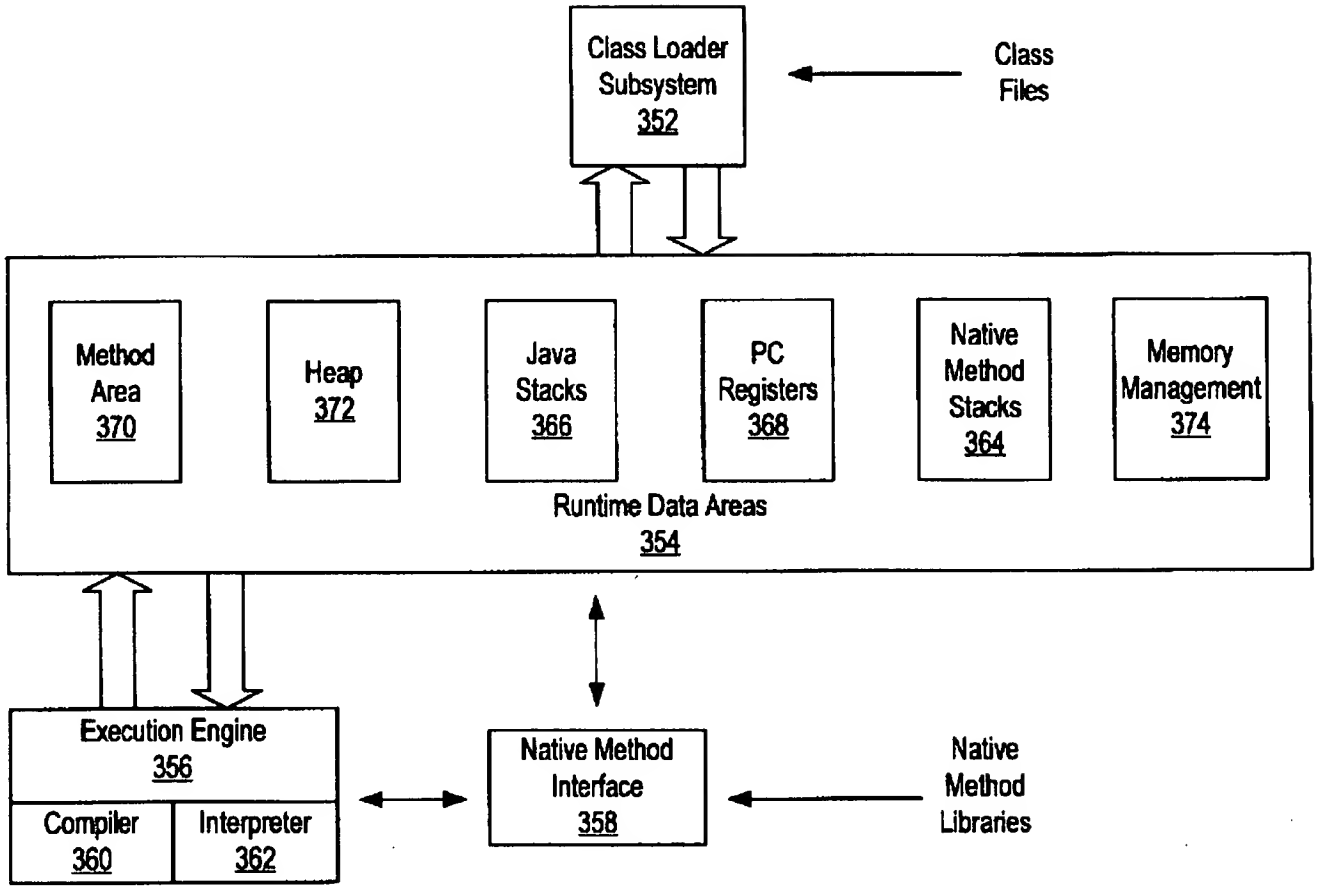
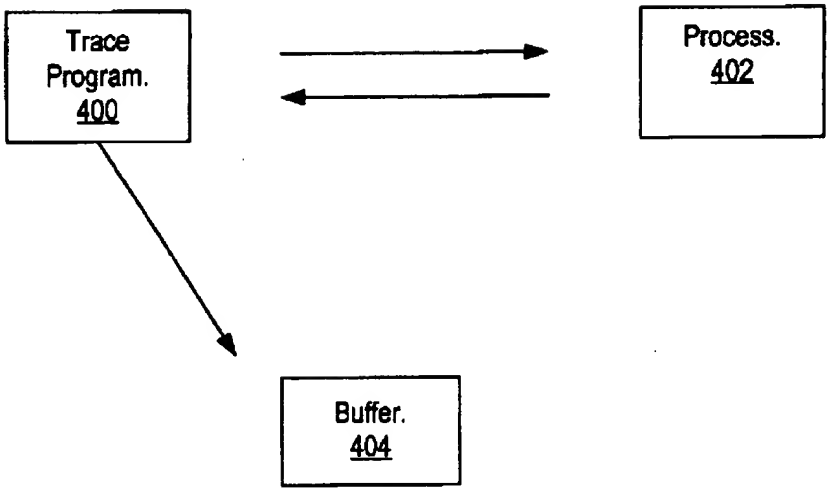


Figure 4
AUS990853



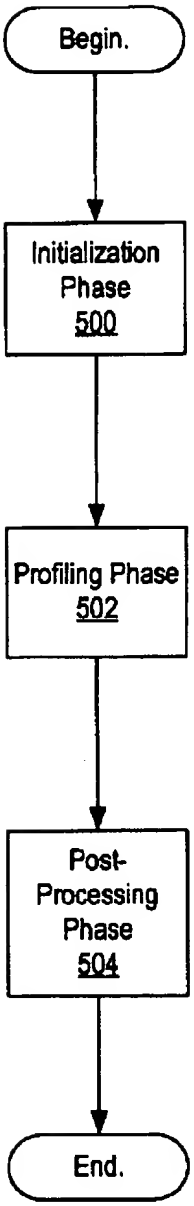


Figure 5

AUS990853US1

Figure 6

AUS990853

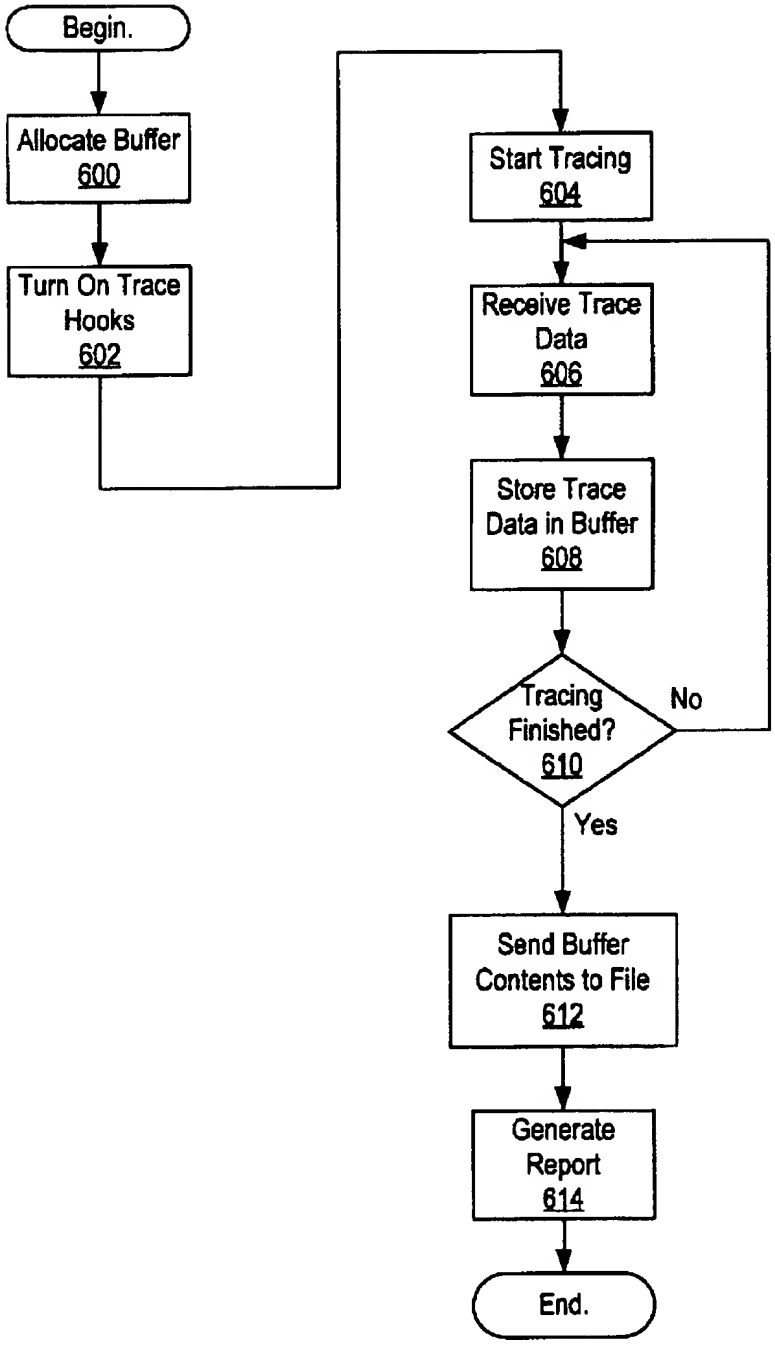


Figure 7

AUS990853

Interrupt Hook

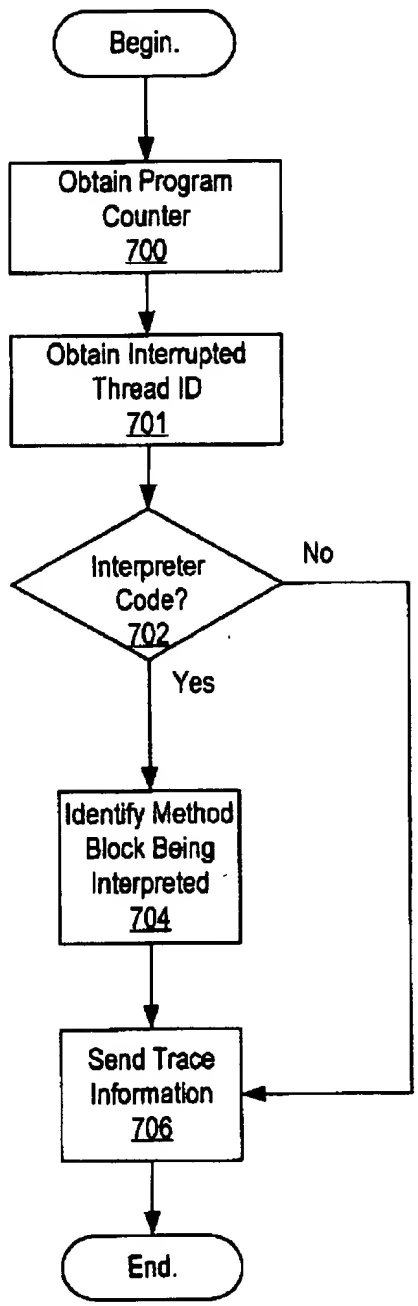


Figure 8

AUS990853

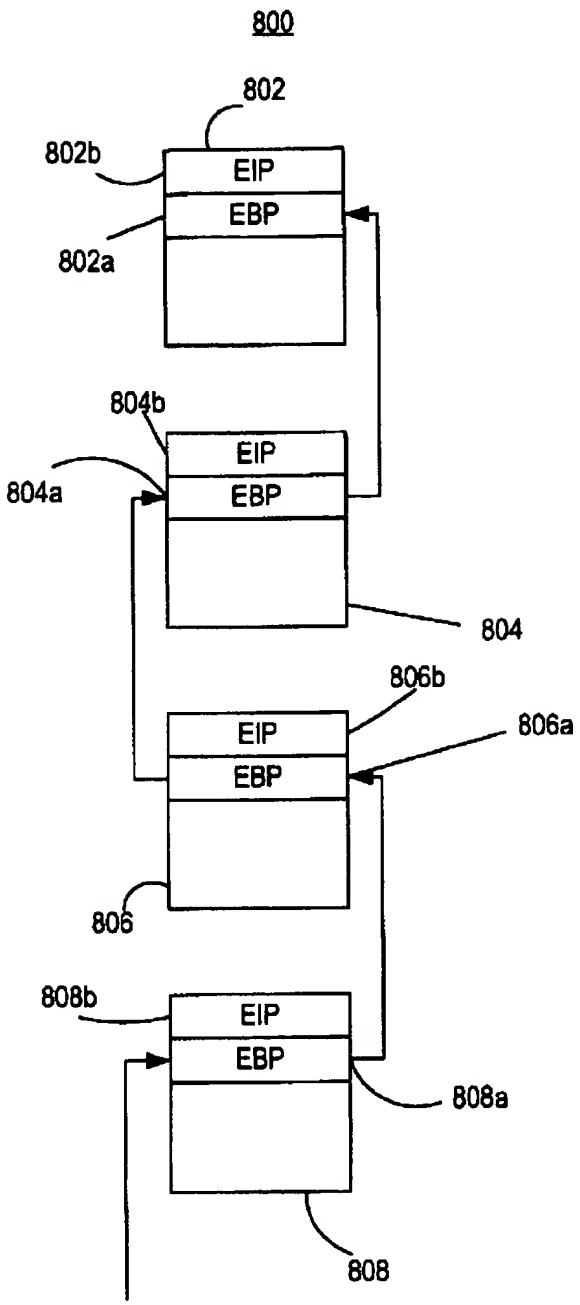
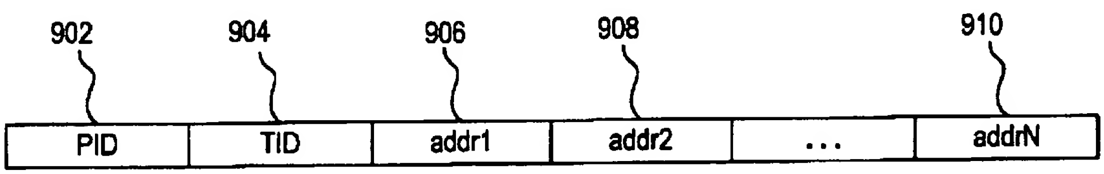


Figure 9

AUS990853

900



Timestamp	Event	Call Stack After
0	enter C	Event
1	enter A	C
2	enter B	CA
3	exit from B	CAB
4	enter B	CA
5	enter B	CAB
6	exit from B	CABB
7	exit from B	CAB
8	exit from A	CA
9	enter B	C
10	enter A	CB
11	enter B	CBA
12	enter A	CBAB
13	exit from A	CBABA
14	exit from B	CBA
15	enter X	CBAX
16	exit from X	CBA
17	exit from A	CB
18	exit from B	C
19	exit from C	

Figure 10A
AUS990853

Sample	Call Stack @ Sample
1	C
2	CAB
3	CAB
4	CAB
5	C
6	CBA
7	CBABA
8	CBA
9	CBA
10	C

Figure 10B
AUS990853

Figure 10C

AUS990853

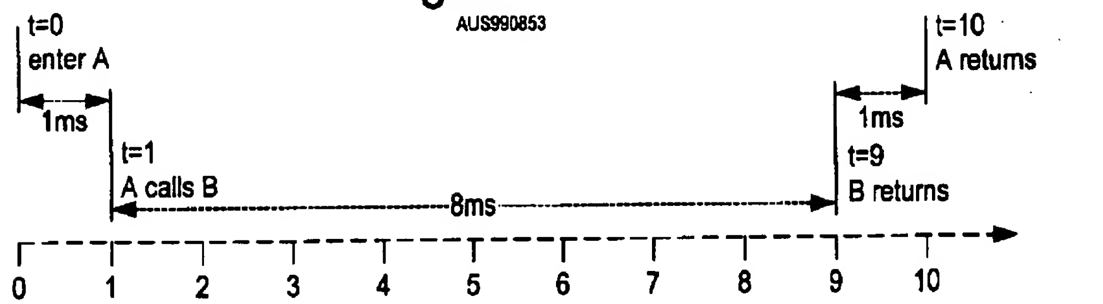


Figure 10D

AUS990853

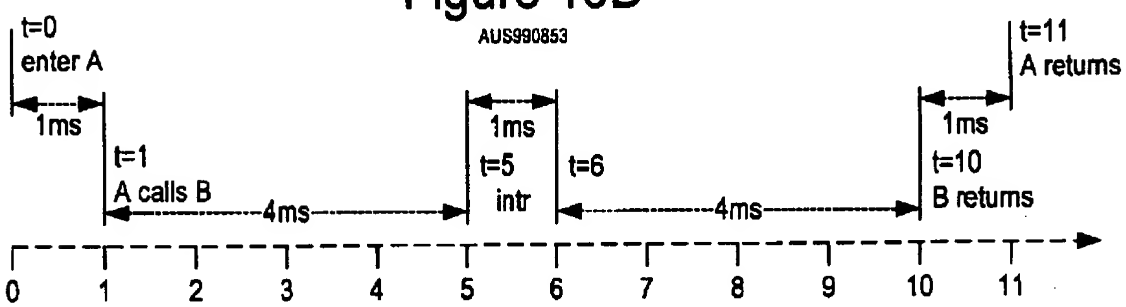


Figure 11A

AUS990853

1100

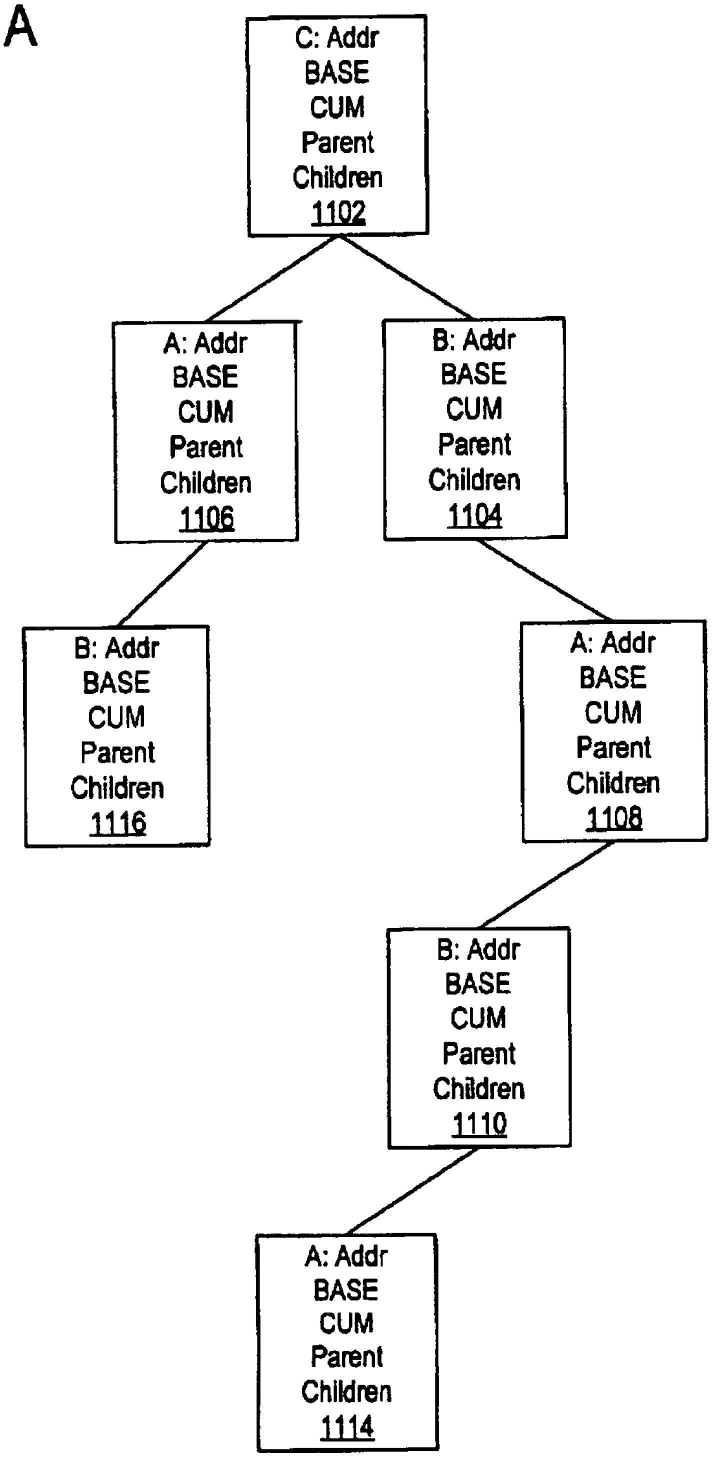


Figure 11B

AUS990853

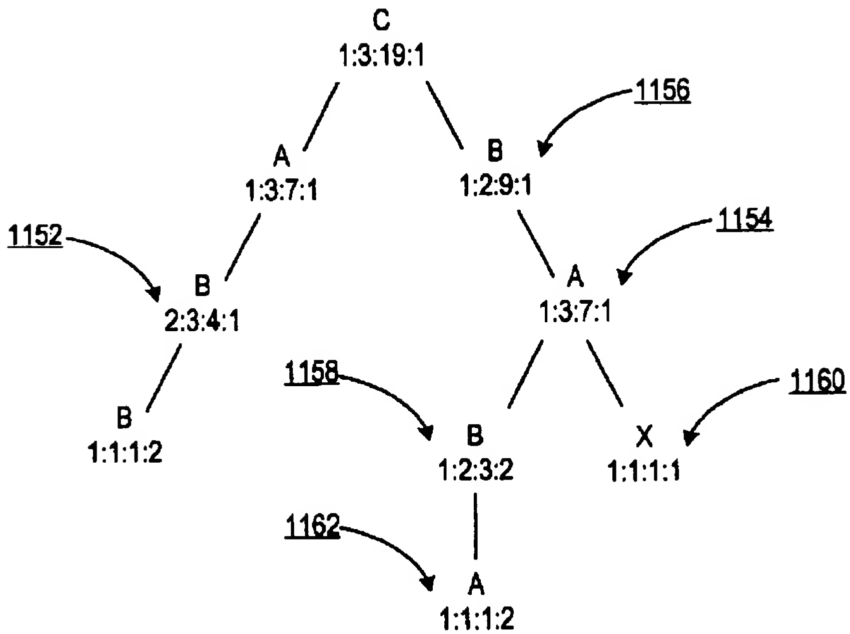


Figure 12

AUS990853

Level 1230	RL 1232	Calls 1234	Base 1236	Cum 1238	Indent 1240
0	1	1	0	19	pt_pidtid
1	1	1	3	19	- C
2	1	1	3	7	- - A
3	1	2	3	4	- - - B
4	2	1	1	1	- - - - B
2	1	1	2	9	- - B
3	1	1	3	7	- - - A
4	2	1	2	3	- - - - B
5	2	1	1	1	- - - - - A
4	1	1	1	1	- - - - X

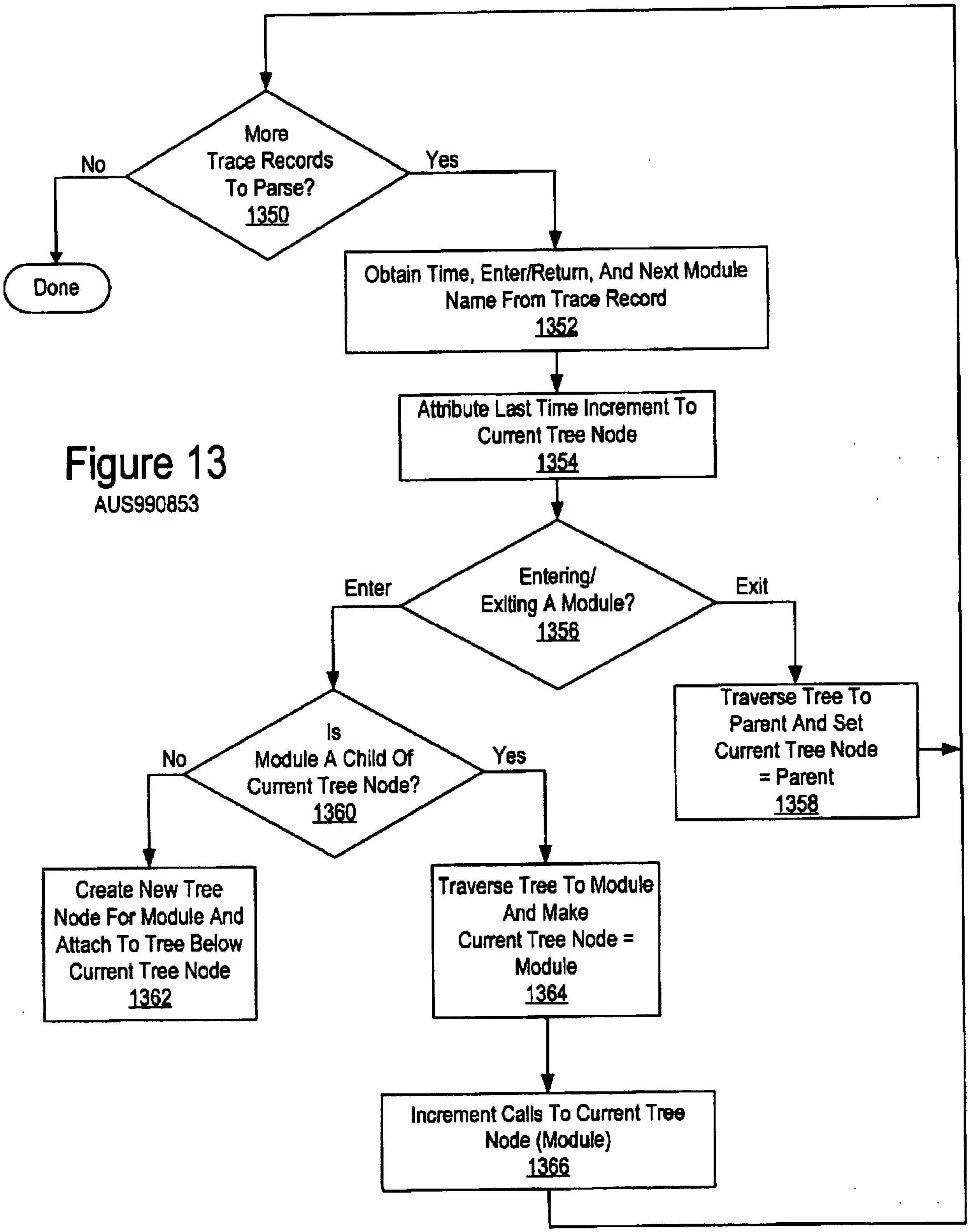

More Trace Records To Parse? 1350
No
Yes
Done
Obtain Time, Enter/Return, And Next Module Name From Trace Record 1352
Attribute Last Time Increment To Current Tree Node 1354
Entering/ Exiting A Module? 1356
Enter
Exit
Traverse Tree To Parent And Set Current Tree Node = Parent 1358
Is Module A Child Of Current Tree Node? 1360
No
Yes
Create New Tree Node For Module And Attach To Tree Below Current Tree Node 1362
Traverse Tree To Module And Make Current Tree Node = Module 1364
Increment Calls To Current Tree Node (Module) 1366

Figure 13

AUS990853

Figure 14
AUS990853

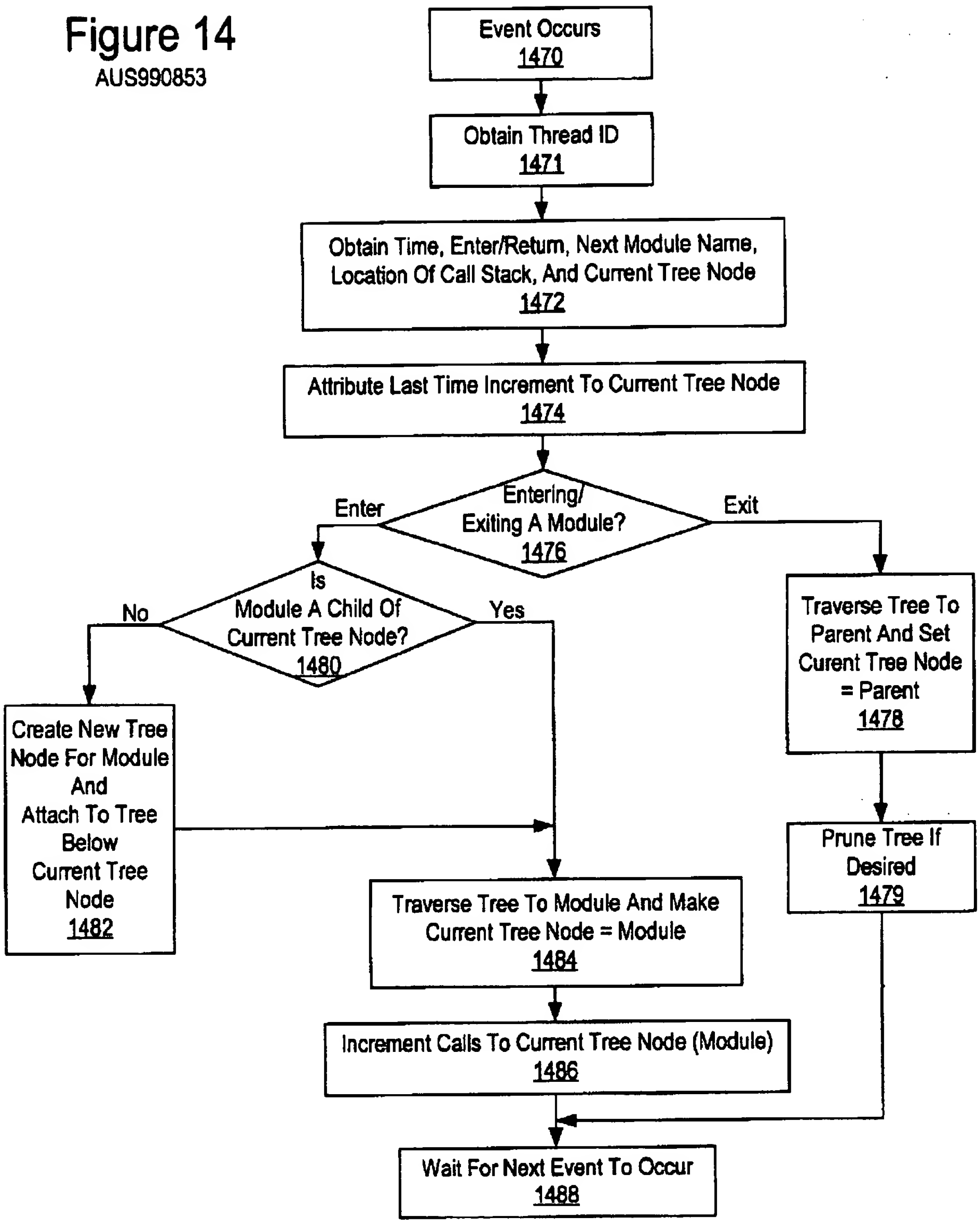
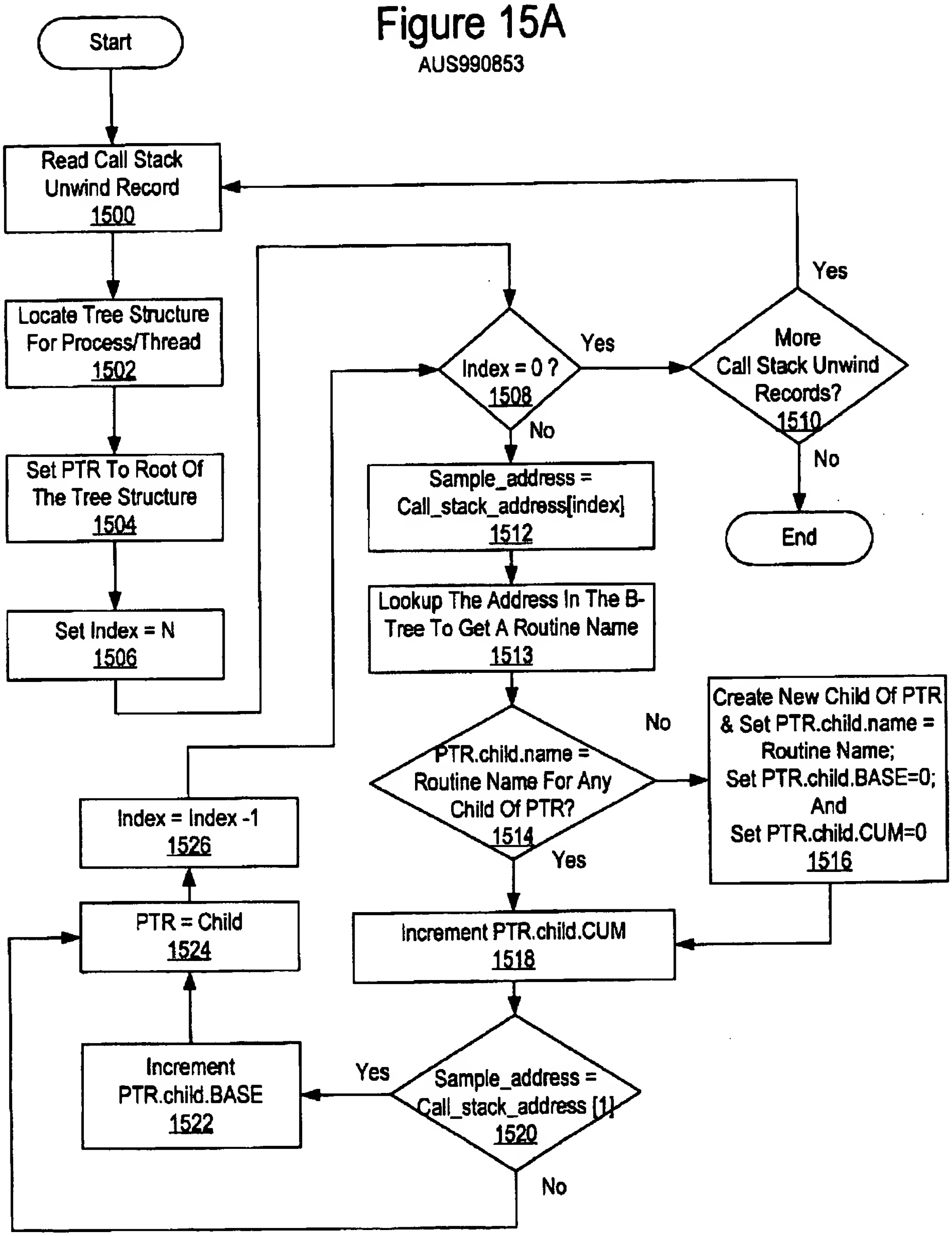

Event Occurs 1470
Obtain Thread ID 1471
Obtain Time, Enter/Return, Next Module Name, Location Of Call Stack, And Current Tree Node 1472
Attribute Last Time Increment To Current Tree Node 1474
Entering/ Exiting A Module? 1476
Enter
Exit
Is Module A Child Of Current Tree Node? 1480
No
Yes
Create New Tree Node For Module And Attach To Tree Below Current Tree Node 1482
Traverse Tree To Module And Make Current Tree Node = Module 1484
Increment Calls To Current Tree Node (Module) 1486
Traverse Tree To Parent And Set Curent Tree Node = Parent 1478
Prune Tree If Desired 1479
Wait For Next Event To Occur 1488

Figure 15A

AUS990853



Start
Read Call Stack Unwind Record 1500
Locate Tree Structure For Process/Thread 1502
Set PTR To Root Of The Tree Structure 1504
Set Index = N 1506
Index = 0 ? 1508
Yes
No
More Call Stack Unwind Records? 1510
Yes
No
End
Sample_address = Call_stack_address[index] 1512
Lookup The Address In The B-Tree To Get A Routine Name 1513
PTR.child.name = Routine Name For Any Child Of PTR? 1514
No
Yes
Create New Child Of PTR & Set PTR.child.name = Routine Name; Set PTR.child.BASE=0; And Set PTR.child.CUM=0 1516
Increment PTR.child.CUM 1518
Sample_address = Call_stack_address [1] 1520
Yes
No
Increment PTR.child.BASE 1522
PTR = Child 1524
Index = Index -1 1526

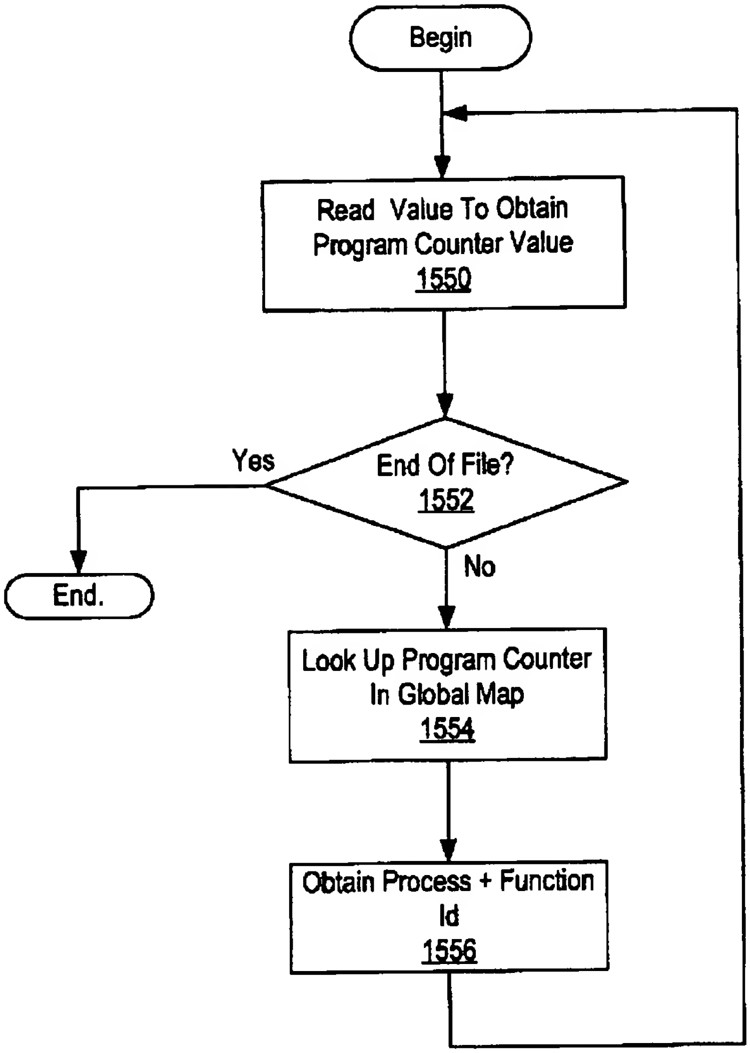


Figure 15B

AUS990853

Figure 16

AUS990853

1600

Calls 1602	Base 1604	Cum 1606	Name 1608
1	0	19	pt_pit.tid
1	3	19	C
3	7	14	A
5	8	13	B
1	1	1	X

Figure 17

AUS990853

<u>1700</u>

ArcFlow Output

Base - Time/Instructions directly in function

Cum - Time/Instructions directly & indirestly in function

ArcFlow Invarients:

1) Sum(Parent(Calls)) = Self(Calls)
2) Sum(Parent(Base)) = Self(Base)
3) Sum(Parent(Cum)) = Self(Cum)
4) Sum(Child(Cum)) = Self(Cum) - Self(Base)

Source	Calls	Base	Cum	Function
Self	1	0	19	[0] pt_pidtid
Child	1	3	19	C
Parent	1	3	19	pt_pidtid
Self	1	3	19	[1] C
Child	1	2	9	B
Child	1	3	7	A
Parent	1	3	7	C
Parent	1	3	7	B
rParent	1	1	1	B
Self	3	7	17 15	[2] A
Child	3	5	7	B
Child	1	1	1	X
Parent	2	3	4	A
rParent	1	2	3	A
Parent	1	2	9	C
Self	5	8	13 17	[3] B
Child	1	3	7	A
rChild	1	1	1	A
Child	1	1	1	B
Parent	1	1	1	A
Self	1	1	1	[4] X

Figure 18

AUS990853

1800

Units :: Ticks
Total :: 342

LvL	RL	Calls	Base	Cum	Indent Name	Ref.
1	1	1	0	342	- _Thread-21__(0xe0046618)	
2	1	3	0	342	-- J:nulltestScore()I	
3	1	2	0	272	--- J:nulltestMilliseconds(I)I	
4	1	29450	0	271	---- J:nullexecute()I	
5	1	271	0	271	----+ stack_0x40	
6	1	271	0	271	----+- F:ExecuteJava	1802
7	1	271	0	271	----+-- F:jit_invokeCompiledEntryMethod	
8	1	271	0	271	----+--- F:_jit_invokeentry	
9	1	271	0	271	----+---- F:JITInvokeCompiledEntryMethod_md	
10	1	271	0	271	----+----+ J:nullrun()V	
11	2	271	0	271	----+----+- J:nulltestScore()I	
12	2	271	0	271	----+----+-- J:nulltestMilliseconds(I)I	1806
13	2	271	268	271	----+----+--- J:nullexecute()I	
14	1	2	0	2	----+----+---- F:jperf_methodEntry	
15	1	2	0	2	----+----+----+ F:SoftTracehook	
16	1	2	2	2	----+----+----+- F:enable_interrupts	
14	1	1	1	1	----+----+---- F:jperf_methodExit	
4	1	1	0	1	---- stack_0x40	
5	1	1	0	1	----+ F:ExecuteJava	1804
6	1	1	0	1	----+- F:jit_invokeCompiledEntryMethod	
7	1	1	0	1	----+-- F:_jit_invokeentry	
8	1	1	0	1	----+--- F:JITInvokeCompiledEntryMethod_md	
9	1	1	0	1	----+---- J:nullrun()V	
10	2	1	0	1	----+----+ J:nulltestScore()I	
11	2	1	0	1	----+----+- J:nulltestMilliseconds(I)I	1808
12	1	1	0	1	----+----+-- J:nullexecute()I	
13	1	1	0	1	----+----+--- F:jperf_methodExit	
14	1	1	0	1	----+----+---- F:SoftTracehook	
15	1	1	1	1	----+----+----+ F:enable_interrupts	
4	1	2	0	0	---- J:nullcleanUp()I	

Figure 19

AUS990853

Major Code	Minor Code	Data Item 1	Data Item 2	Data Item 3	Data Item 4	Data Item 5	Description
0x4	level + 0x1	depth	n/a	n/a	n/a		begin interrupt at level
0x4	level + 0x80000001	depth	n/a	n/a	n/a		end interrupt at level
0x10	0xab	system tid	java tid	is System Thread (boolean)	n/a		thread created without a name while trace active
0x10	0xac	system tid	n/a	n/a	n/a		identifies the idle thread
0x10	0xad	system tid	n/a	n/a	n/a		identifies the garbage collection thread
0x10	0xae	system tid	java tid	thread name	n/a		thread created with a name while trace active
0x30	0x10	object id	method block address	n/a	n/a		method invocation (interpreted)
0x30	0x10 + 0x80000000	object id	method block address	n/a	n/a		method exit (interpreted)
0x40	0x7fffffff	number (n) of stack unwinds at timer interrupt	pc1-program counter of interrupted routine	pc2-caller of interrupted routine	....	pcn-1 of n-2nd caller of interrupted routine	pcn of n-1st caller of interrupted routine
0x41	0x7fffffff	number (n) of stack unwinds at instrumented routine	pc1-program counter of instrumented routine	pc2-caller of instrumented routine	....		pcn of n-1st caller of instrumented routine
0x50	0x10	object id	method block address	n/a	n/a		method invocation (jitted)
0x50	0x10 + 0x80000000	object id	method block address	n/a	n/a		method exit (jitted)

Figure 20

AUS990853

Application Level Thread Oriented Approach

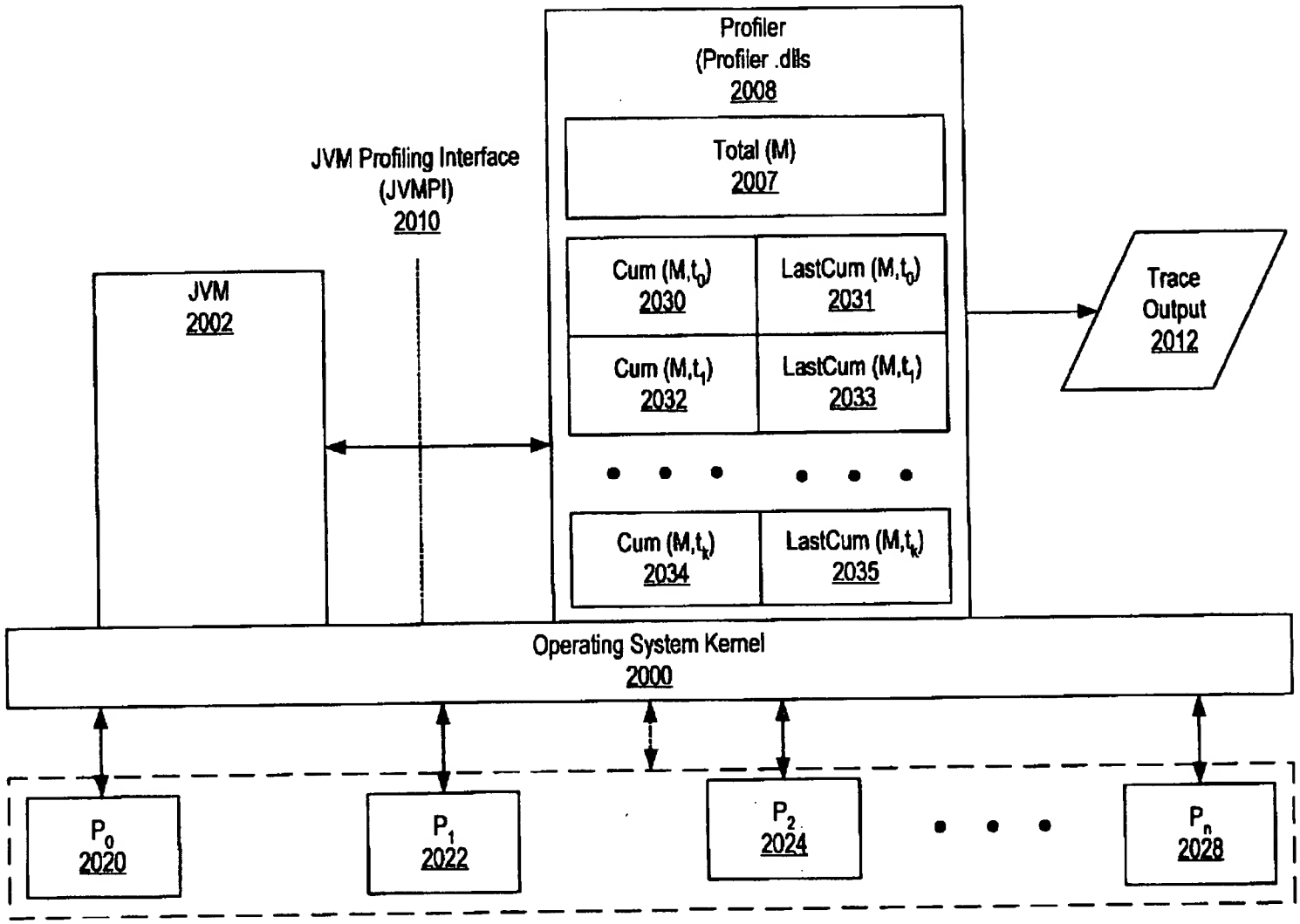


Figure 21A

AUS990853

Application Level

Thread Oriented Approach

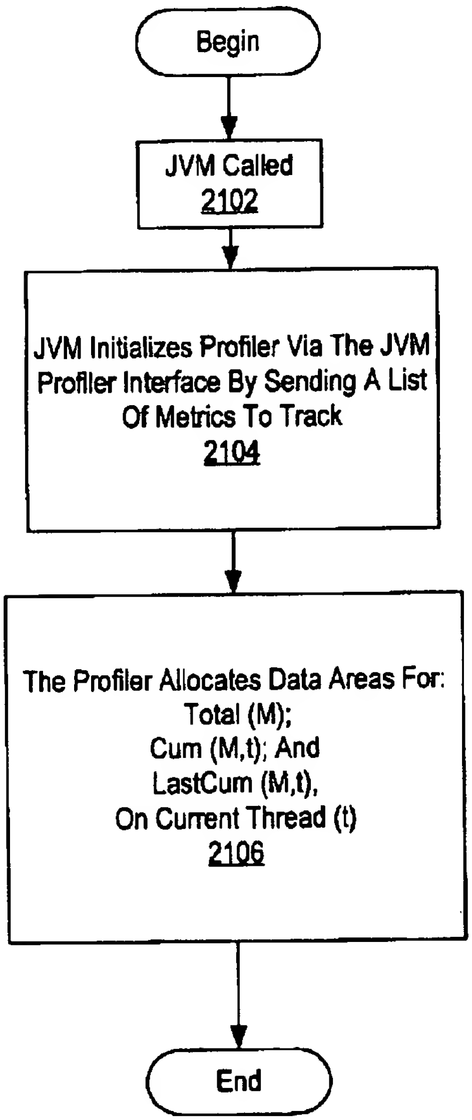


Figure 21B

AUS990853

Application Level

Thread Oriented Approach for Each New Thread

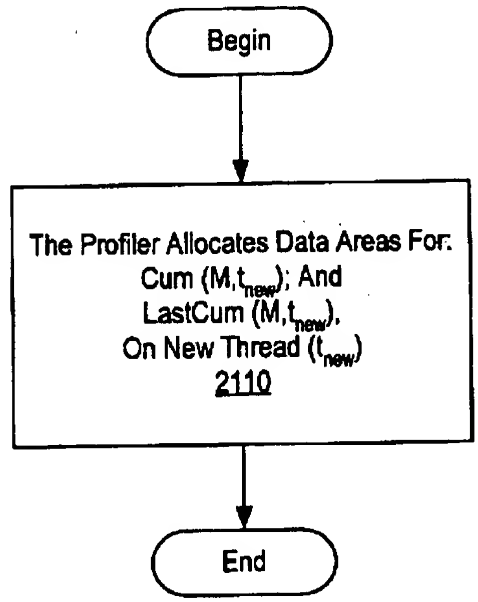


Figure 22

AUS990853

Profiler Receives Metric Event from Jvm

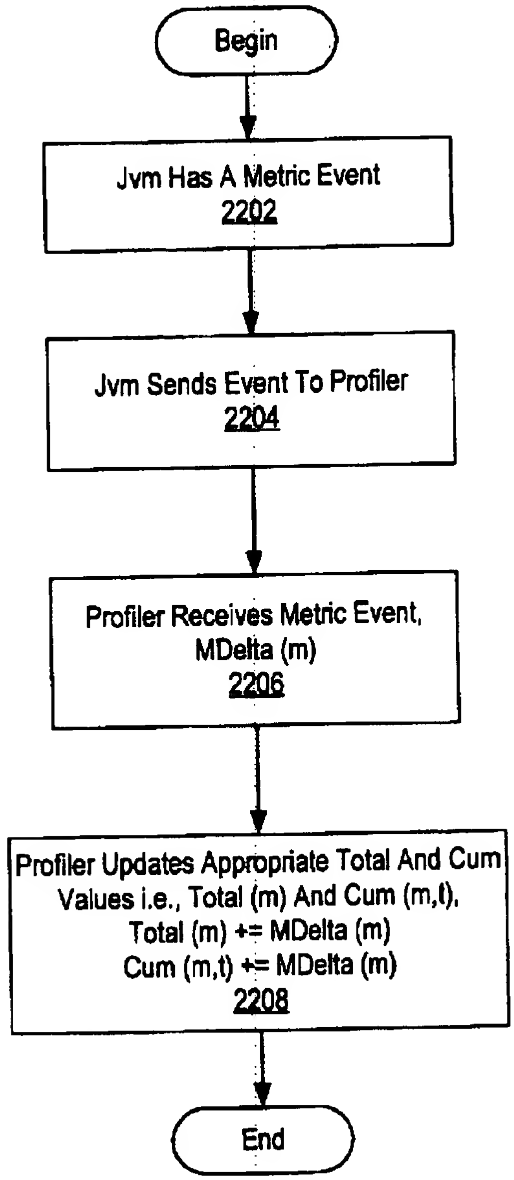


Figure 23

AUS990853

<u>Application Level Thread Oriented Approach</u>

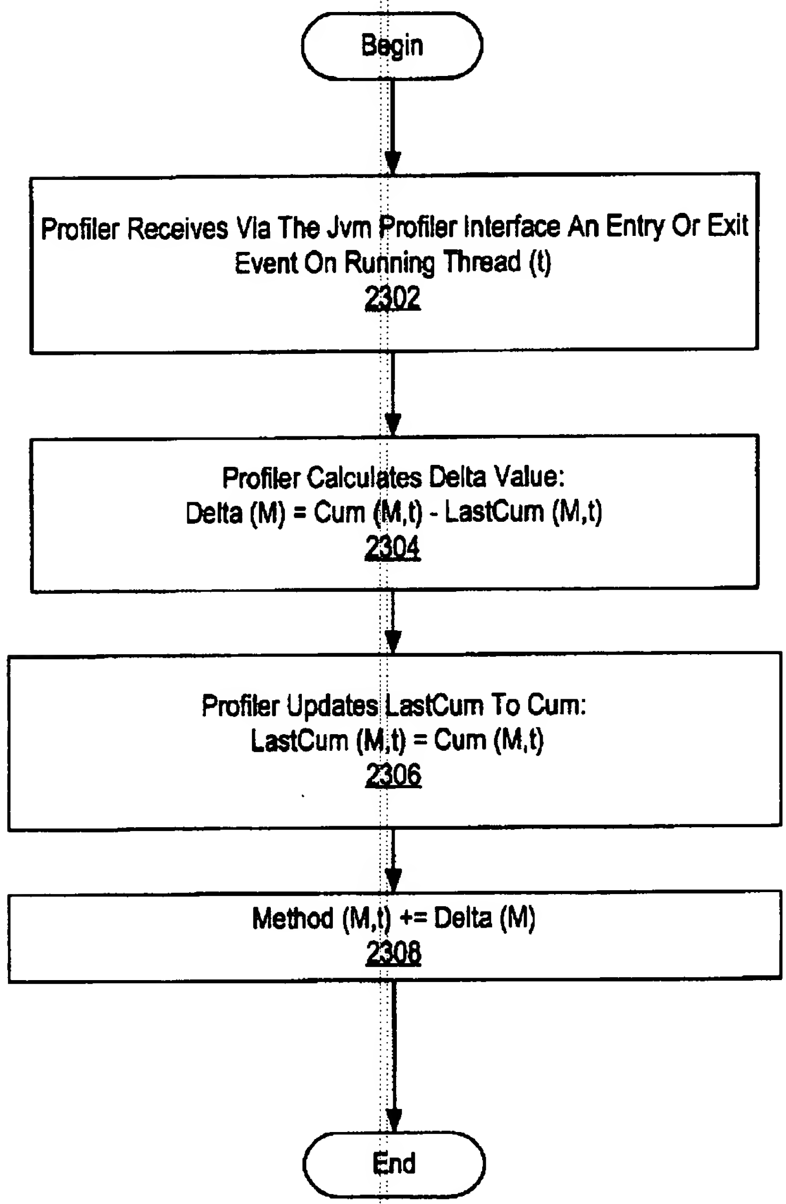


Figure 24

AUS990853

API for Metric Values

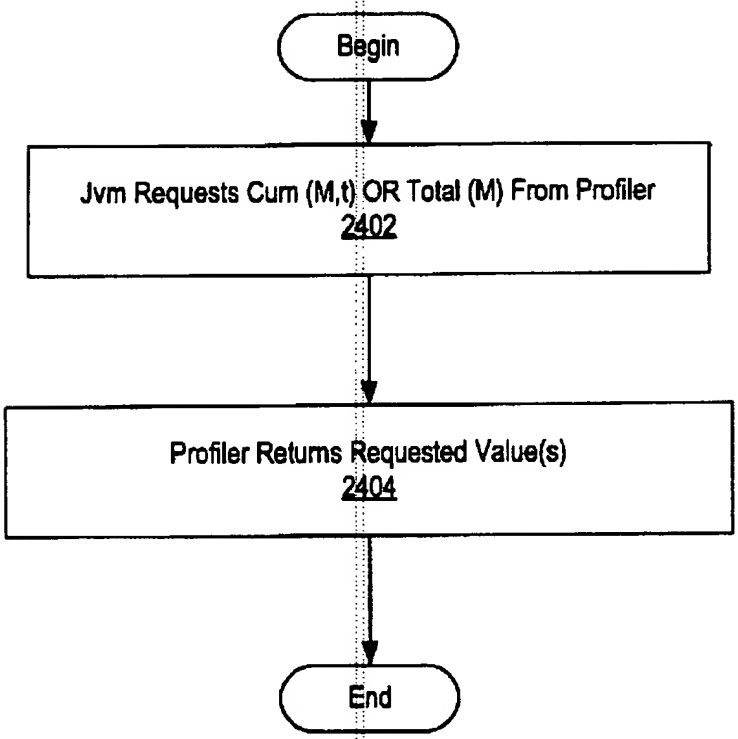


Figure 25

AUS990853

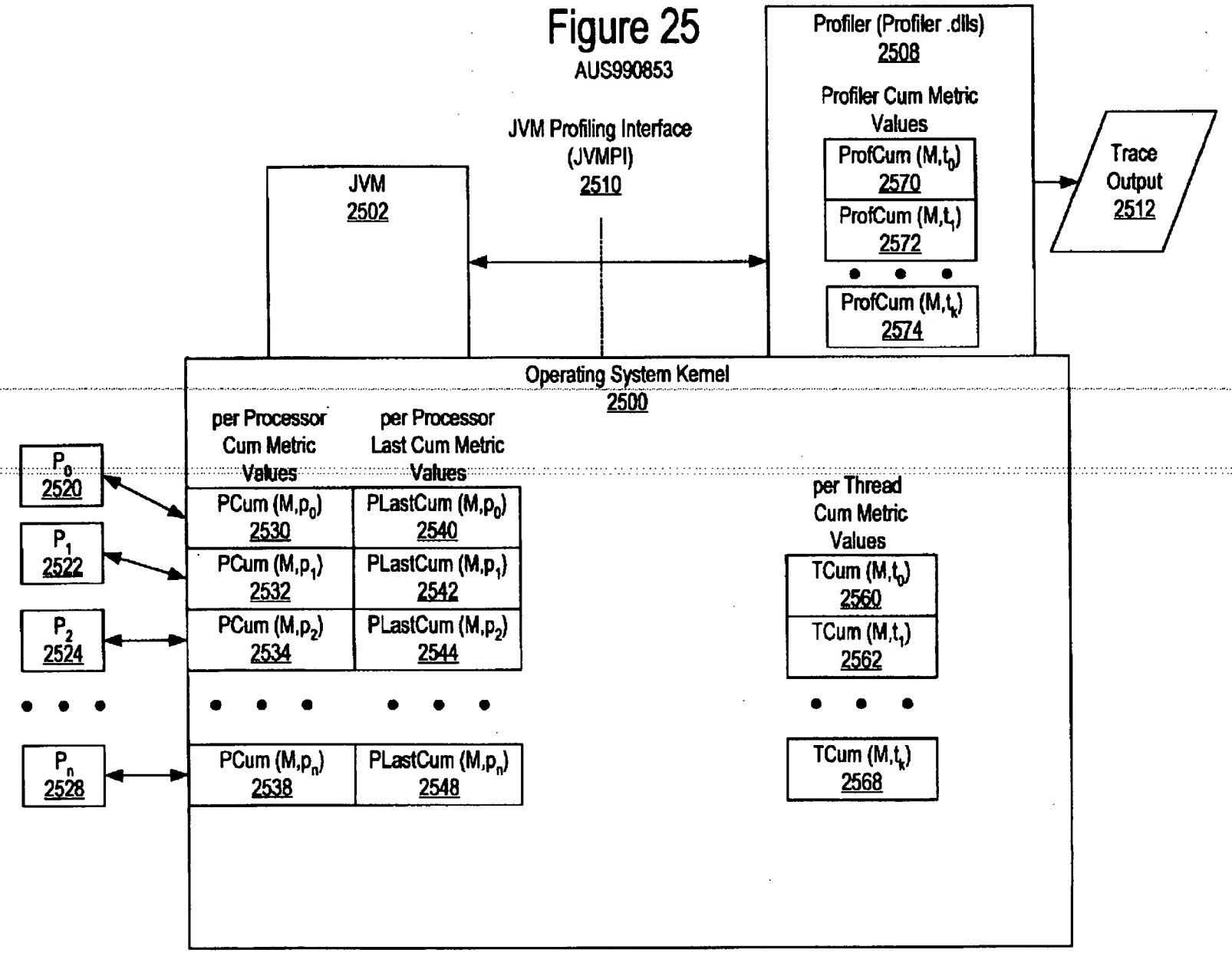


Figure 26A

AUS990853

Initialization Process

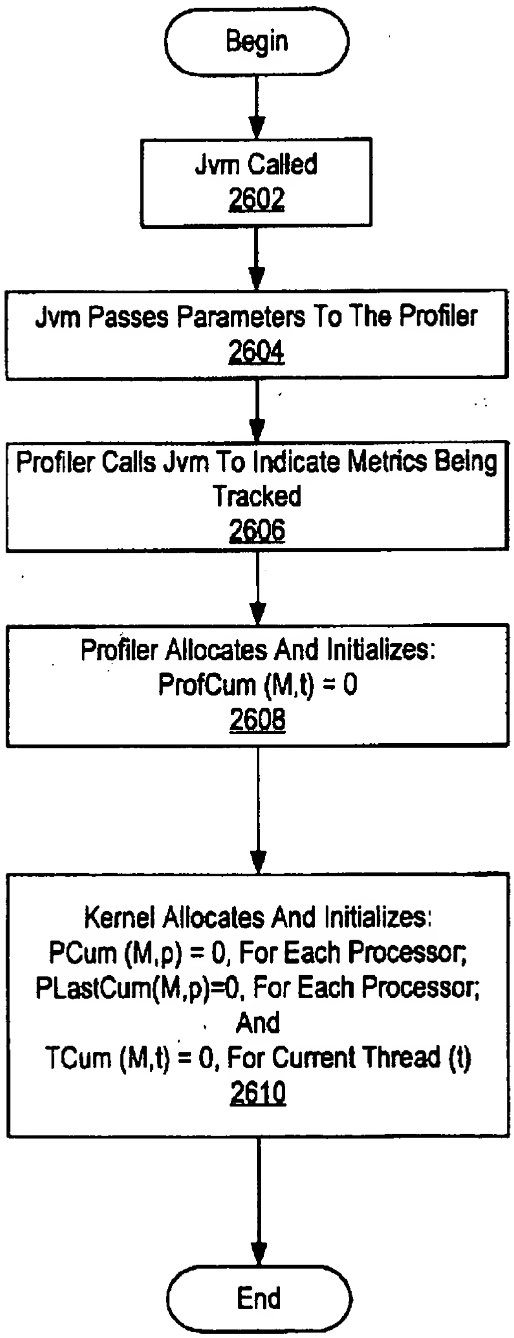

Begin
Jvm Called
2602
Jvm Passes Parameters To The Profiler
2604
Profiler Calls Jvm To Indicate Metrics Being Tracked
2606
Profiler Allocates And Initializes:
ProfCum (M,t) = 0
2608
Kernel Allocates And Initializes:
PCum (M,p) = 0, For Each Processor;
PLastCum(M,p)=0, For Each Processor;
And
TCum (M,t) = 0, For Current Thread (t)
2610
End

Figure 26B

AUS990853

Process for the Kernel Recognizing a New Thread on Processor (p)

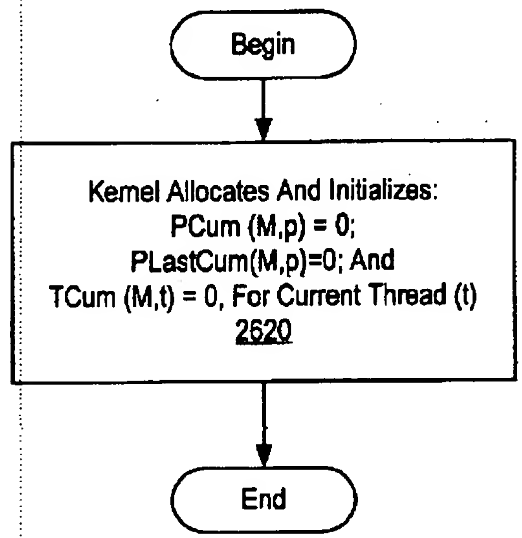


Figure 26C

AUS990853

Process for the Profiler Recognizing a New Thread

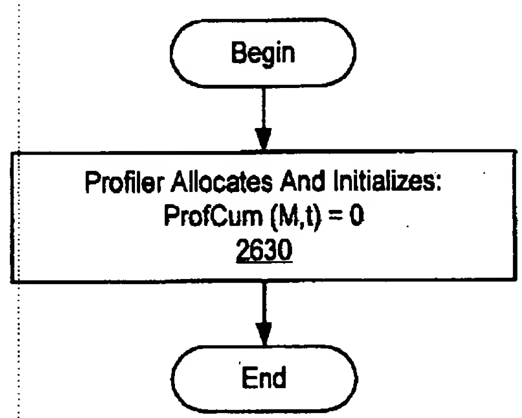


Figure 27

AUS990853

Jvm Process for Updating Variable Values for Metrics Being Tracked

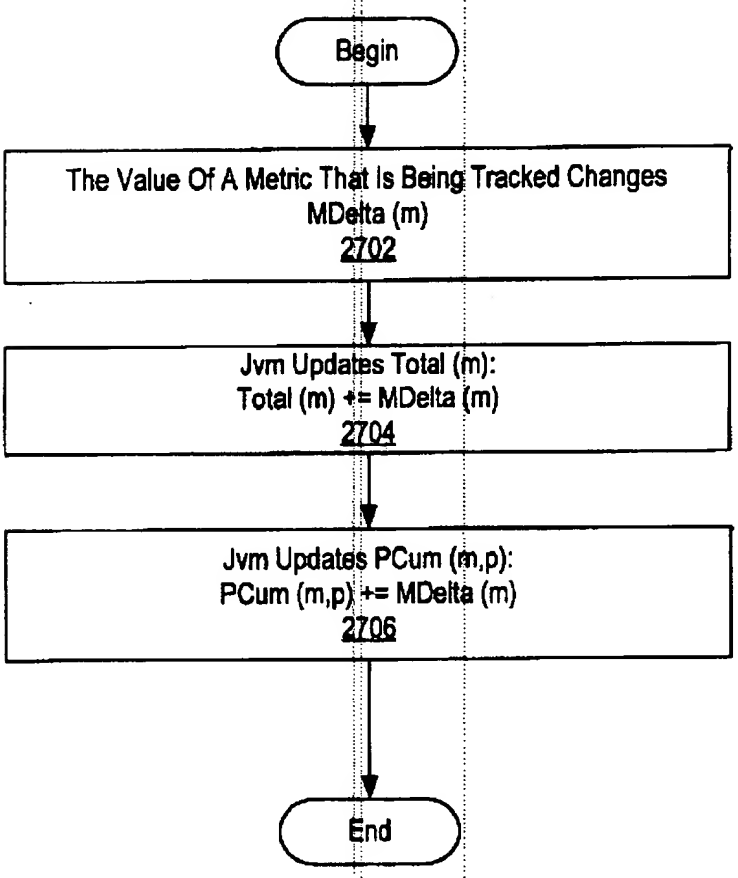


Figure 28

AUS990853

Process for the Jvm Directly Updating Per Processor Global Variable Values for Metrics Being Tracked

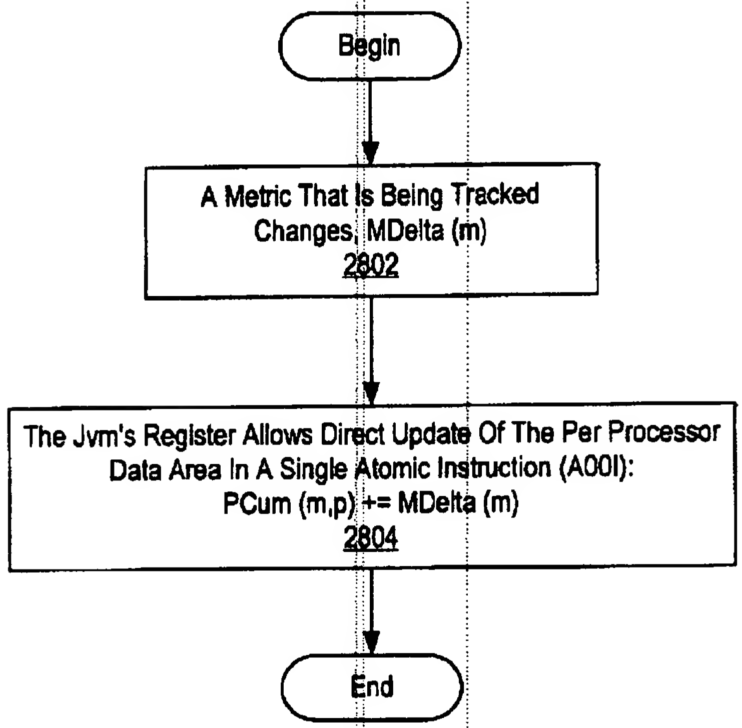


Figure 29

AUS990853

Process for the OS Kernel Updating Per Processor Global Variable Values for Metrics Being Tracked

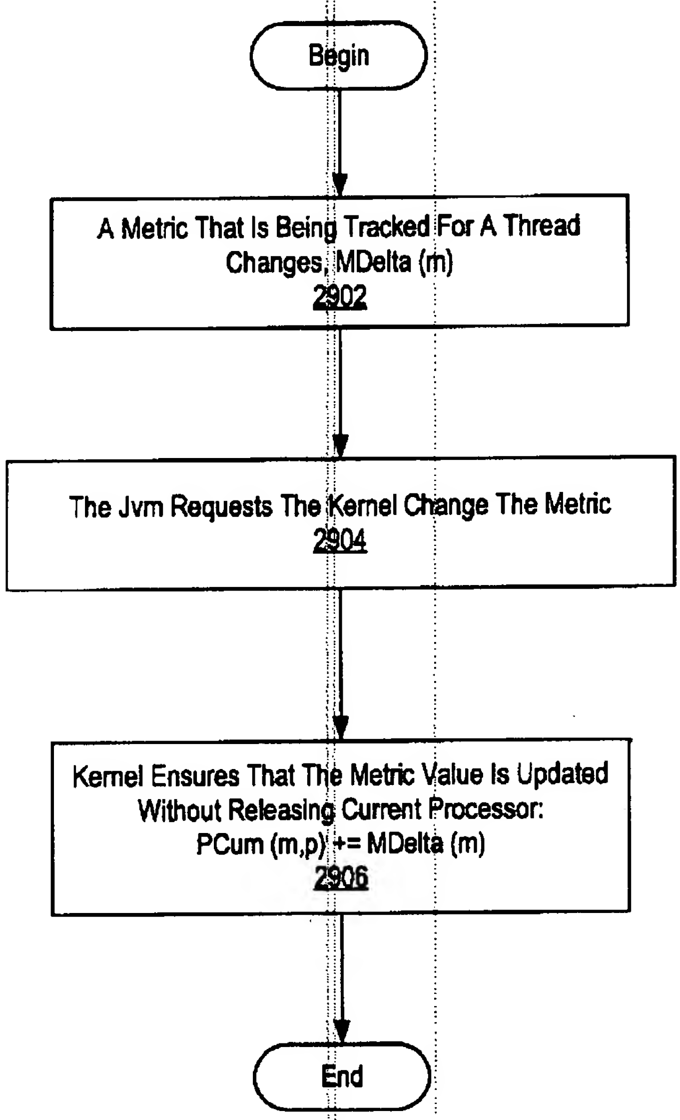


Figure 30

AUS990853

Process for the Kernel Updating Base Metric Variable Values for Metrics Being Tracked in Response to a Thread Dispatch Event

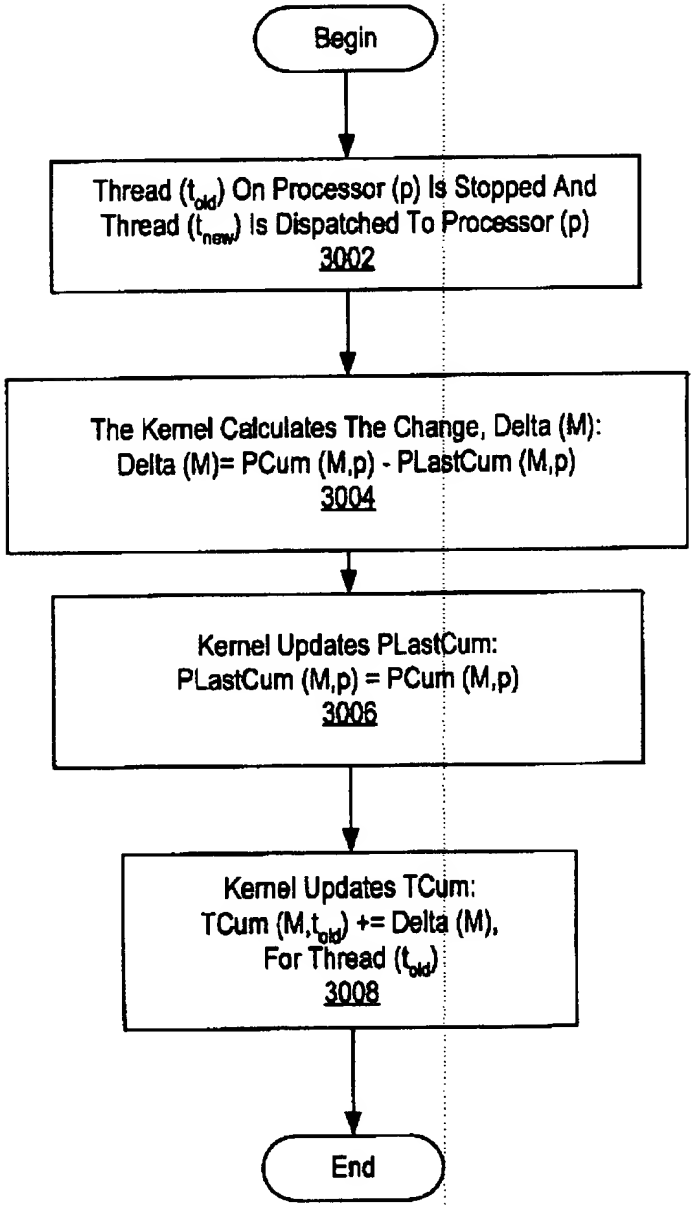
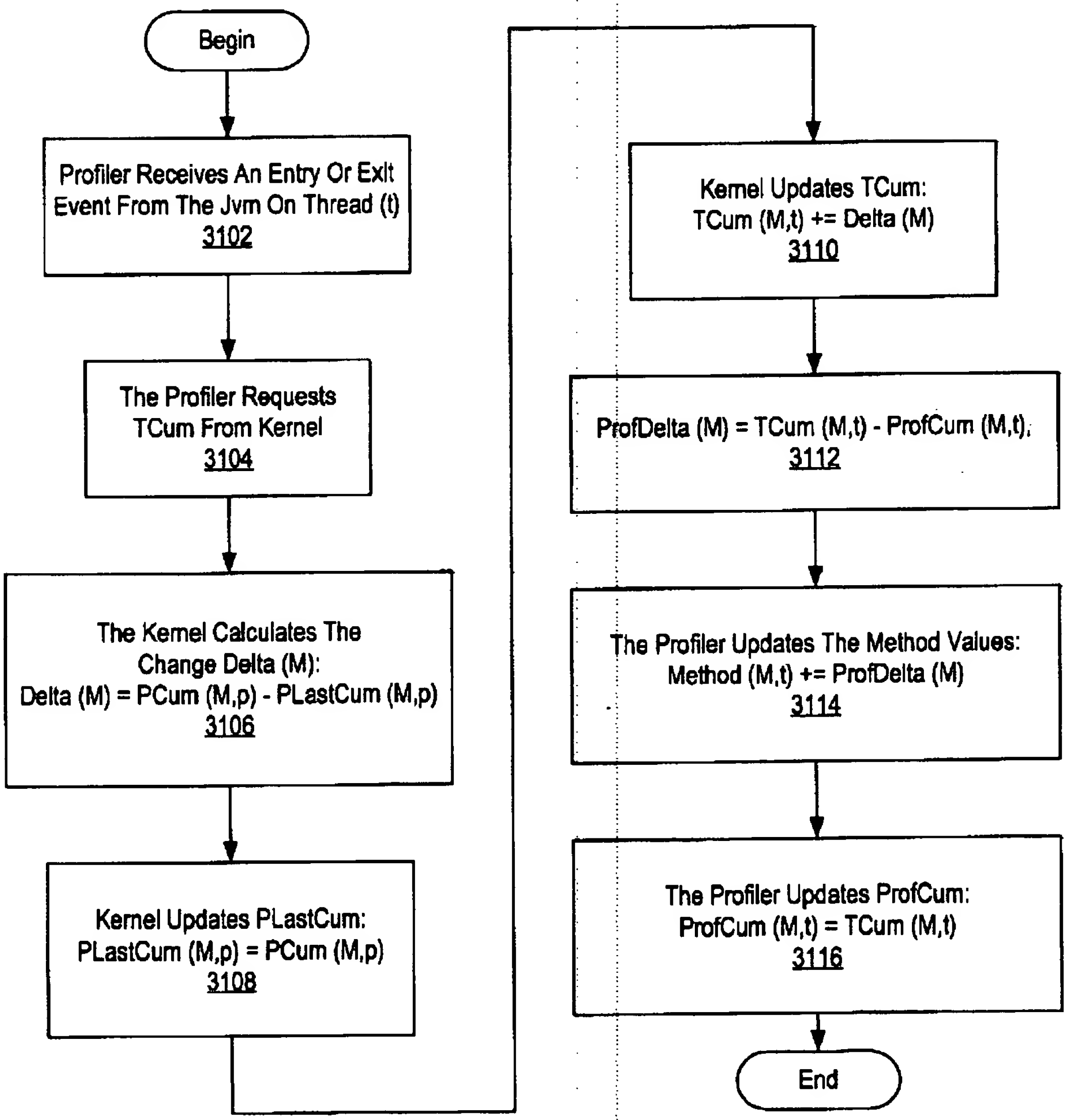


Figure 31

AUS990853

Process for the Profiler Updating Base Metric Variable Values for Metrics Being Tracked in Response to a Method Entry or Exit Event



Begin
Profiler Receives An Entry Or Exit Event From The Jvm On Thread (t) 3102
The Profiler Requests TCum From Kernel 3104
The Kernel Calculates The Change Delta (M): Delta (M) = PCum (M,p) - PLastCum (M,p) 3106
Kernel Updates PLastCum: PLastCum (M,p) = PCum (M,p) 3108
Kernel Updates TCum: TCum (M,t) += Delta (M) 3110
ProfDelta (M) = TCum (M,t) - ProfCum (M,t) 3112
The Profiler Updates The Method Values: Method (M,t) += ProfDelta (M) 3114
The Profiler Updates ProfCum: ProfCum (M,t) = TCum (M,t) 3116
End